Graphes en OCaml

OPTION INFORMATIQUE - TP n° 2.9 - Olivier Reynet

A Représentation

A1. Quel est le type OCaml du graphe suivant? `let g = [| [1;2] ; [0;3;4] ; [0;5;6] ; [1] ; [1] ; [2] ; [2] |]`. Le dessiner.

A2. Écrire une fonction de signature `liste_vers_matrice : int list array -> bool array array` qui tranforme une liste d'adjacence en matrice d'adjacence.

A3. Écrire une fonction de signature `matrice_vers_liste : bool array array -> int list array` qui effectue l'opération inverse.

B Parcours de graphe en mode récursif

B4. Écrire une fonction de signature `parcours_largeur : int list array -> int -> int list` qui parcours en largeur un graphe donné sous la forme d'une liste d'adjacence. Cette fonction opère à l'aide d'une fonction récursive auxiliaire qui explore les sommets dans le bon ordre. Un type file d'attente n'est **pas** nécessaire, une liste et l'opérateur `@` suffisent. Pour mémoriser les sommets découverts on utilisera un tableau de booléens ou une liste de sommets découverts. Comparer la complexité des deux approches.

B5. Écrire une fonction de signature `parcours_profondeur : int list array -> int -> int list` qui parcours en profondeur un graphe donné sous la forme d'une liste d'adjacence. Cette fonction opère à l'aide d'une fonction récursive auxiliaire qui explore les sommets dans le bon ordre. La structure de pile n'est pas nécessaire, une liste suffit. Pour mémoriser les sommets découverts on utilisera un tableau de booléens. On pourra utiliser la fonction `List.fold_left`